THE UNIVERSITY OF NEW MEXICO

ECE 538 Advanced Computer Architecture

Instructor: Lei Yang

Department of Electrical and Computer Engineering

October 25, 2021

Review
Mid-Term Exam

PERFORMANCE: METRICS & ENHANCEMENT TECHS

- Factors affect execution time?
  - $\boldsymbol{Execution\ Time = IC \times CPI \times T}$
  - Instruction count (*IC*): Application/program; ISA
  - Clock per instruction (*CPI*): ISA; Datapath design; Parallel and pipeline HW design
  - Clock period (*T*): Semiconductor technology; Datapath design and implementation

- Decrease in one may lead to increase in other two.

PERFORMANCE: METRICS & ENHANCEMENT TECHS

- Challenges on performance enhancement
  - Reduction of clock cycle time (T) / increase clock frequency
    - Power consumption increases with increase in clock frequency
    - Memory operations take longer than a clock period->***memory-wall problem***(memory is relatively slower than CPU, CPU wait for data/instructions)
  - Reduction of instruction count (IC)
    - More complex instructions
    - Multi-issue processor: VLIW/superscalar, SIMD, vector processor
  - Reduction of cycle per instruction (CPI)
    - Instruction pipelining: Pipeline datapath
    - Multi-issue processor: VLIW/superscalar processor

Performance Speedup

- Case 1: speedup of computer-A over computer-B can be computed as

$$Speedup = \frac{Perf_A}{Perf_B}, Speedup = \frac{Time_B}{Time_A}$$

- Case 2: speedup achieved due to enhancement technique can be as

$$Speedup = \frac{T_{unenhanced}}{T_{enhanced}} = \frac{T_{original}}{T_{enhanced}}$$

Performance Speedup

- Example: If fraction E of the program is enhanced by a factor of S in an enhanced machine, then determine the speedup of enhanced machine over the machine before enhancement (original machine). What is the maximum speed up for a given E if S can be any value greater than or equal to 1?

Fraction $U = (1 - E)$ of the program is not enhanced.

if T is execution time of program in unenhanced (original) machine:

- Time required for the execution of unenhanced fraction $= T\times (1 - E)$
- Time required for the execution of enhanced fraction $= [T\times E]/S$
- Total execution time in the enhanced machine

$$T' = [T\times(1 - E)] + [T\times E]/S$$

$T' = T\times\left[(1 - E) + \frac{E}{S}\right] = T\times[1 - E(S - 1)/S]$ (check: if $S = 1$ then $T' = T$).

Speed up = $\frac{T}{T'} = \frac{T}{T}\times\left[(1 - E) + \frac{E}{S}\right] = \frac{1}{[(1-E)+E/S]}$

Performance Speedup - Amdahl's Law

- Amdahl's law: **the performance improvement to be gained from using some faster mode of execution is limited by the fraction of the time the faster mode can be used.**

  - If fraction (1 – E) of application cannot be enhanced for parallel implementation, then speedup is limited by a factor of 1/(1-E), even if the rest of the application is infinitely sped up, and involve infinitesimal time for the computation.

  - Amdahl's law defines the ***speedup*** that can be gained by using a particular feature. What is the ***speedup***?

$$Speedup = \frac{Perf.of\ entire\ task\ using\ enhancement\ when\ possible}{Perf.of\ rntire\ task\ without\ using\ enhancement}$$

Performance Speedup - Amdahl's Law

- Amdahl's law gives us a quick way to find the speedup from some enhancement, which depends on two factors:
  - The fraction of the computation time in the original computer that can be converted to take advantage of the enhancement.
  - The improvement gained by the enhanced execution mode, that is, how much faster the task would run if the enhanced mode were used for the entire program.

E is fraction of program , enhanced by a factor S.

$$Execution\ tim_{enhanced} = Execution\ tim_{unenhanced} \times (1 - E + \frac{E}{S})$$

$$Thus, \qquad Speedup_{overall} = \frac{Execution\ tim_{unenhanced}}{Execution\ tim_{enhanced}} = \frac{1}{1 - E + \frac{E}{S}}$$

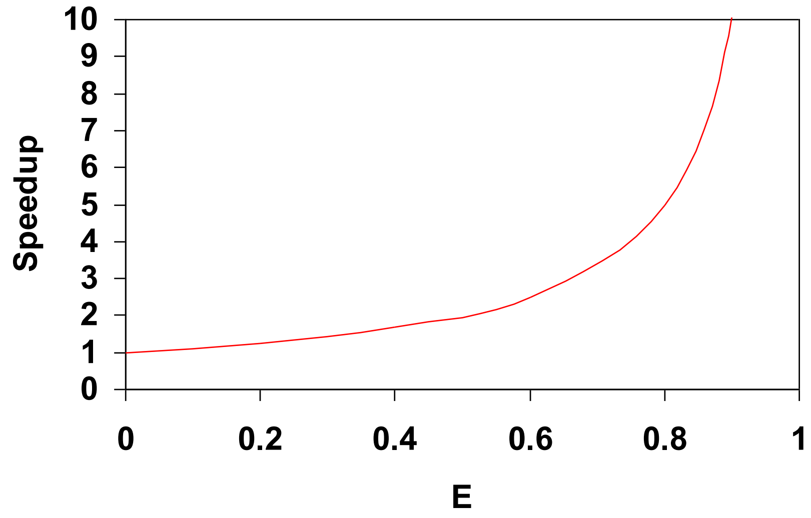
PERFORMANCE SPEEDUP - AMDAHL'S LAW

- Amdahl's law gives us a quick way to find the speedup from some enhancement, which depends on two factors:

E is fraction of program , enhanced by a factor S. Make common case fast:

$$\lim_{s \to \infty} \frac{1}{1 - E + \frac{E}{S}} = \frac{1}{1 - E}$$

$$Speedup_{max} = \frac{1}{1 - E}$$



POWER AND ENERGY

- There are NOT the same metric
  - Energy is measured in Joules (J)
  - Power is J/s
- Energy per task is better metric for efficiency
  - Relate to battery life in personal mobility device(PMDs)
  - Reduce energy bills in WSC!!!!
  - If processor A consumes 2x the power as processor B but complete the same task in the fourth of the time, there is a 2x gain in energy efficiency!

POWER AND ENERGY

Power: We do care about

- Power limitations
  - Must get power **into** the IC and distribute it
  - And **Out** in the form of heat, e.g, a $2.25cm^2$ die could consume 100W

- Thermal Design Power (TDP)
  - Characterizes sustained power consumption
  - Used as target for power supply and cooling system
  - Lower than peak power (1.5x is typical)
  - Must be higher than average!

Power and Energy

Energy Consumption in CMOS

- Dynamic vs. Static
  - Dynamic: switching of transistors and clock
  - Static: current leakage through imperfect transistors
- Dynamic energy
  - Consider transistor switching from 0->1 or 1->0
  - A transistor gate looks like a capacitor
  - Energy in a capacitor is $\frac{1}{2} \times C \times V^2$
- Dynamic power
  - $P_{dynamic} = \frac{1}{2} \times C \times V^2 \times \alpha \times f$ (α is activity factor to account switch/clock cycle)
- Static power
  - $P_{static} = V \times I_{leakage}$ ($I_{leakage}$ depends on # of transistors and their leakage)

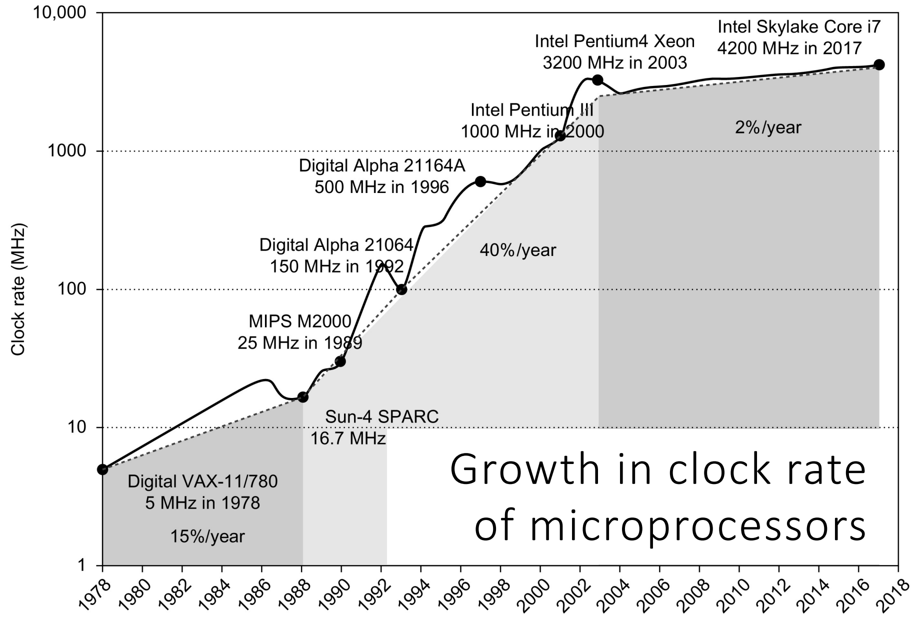
POWER AND ENERGY

Power Observations

- Dynamic power is linearly related to the clock rate and capacitance but quadratically related to voltage
- Static power is linearly related to leakage current and voltage
- Dennard's scaling implied that *C* and *V* of each transistor would scale down as density and *f* go up

- **Power Examples**

- Intel 80386 consumed 2W at 16MHz
- Intel Core i7-6700K consumed 95W at 4.0GHz
- i7 die is 1.5cm x 1.5cm (2.25cm2)
- Reaching the limits of what can be cooled by air

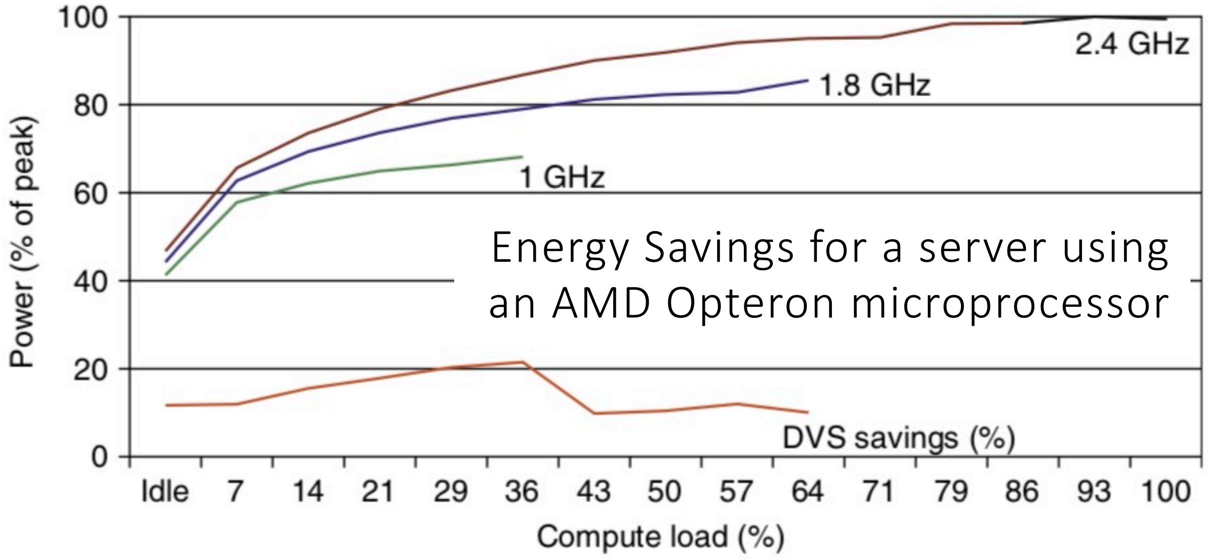


Growth in clock rate of microprocessors

Reducing Power and Energy

- Lower *f* reduces power but not energy per task! Why?
- Significant power savings can be achieved by reducing *V*. How do we do that?
  - Before Dennard's scaling, moving to a new tech node would get us a lower voltage
  - Now days, slowing f down allows V to be lower. Why?
  - This technique is called **Dynamic Voltage Frequency Scaling** (DVFS)

DVFS on server with AMD Opteron



Energy Savings for a server using an AMD Opteron microprocessor

Reducing Power and Energy

- **Clock gating**
  - Turn off the clock of the portions of unused logic
  - The clock network consumes a significant amount of power
  - Eliminates dynamic power for unused logic
  - Static power is still an issue
- **Power gating**
  - Turn off power of portions of unused logic
  - Eliminates both static and dynamic power
  - Dark Silicon!
- **Hardware accelerate**: moving computationally intense portions of software into more efficient dedicated hardware
- **Change state**: put unused cache and memory into a drowsy state where it retains contents but consumes less power by reduced voltages

OTHER PERFORMANCE METRICS

- MTTF: Mean Time to Failure
  - MTTF is the length of time a device or other product is expected to last in operation
  - MTTF is one of the many ways to evaluate the reliability of pieces of hardware or other technologies

Examples

-- MTTF is (perhaps) 100,000 hours for a fan

-- MTTF is (perhaps) 1,000,000 hours for a hard disk; then the failure rate is "1/1000000" per hour

$$FailureRate = \frac{1}{M} = \frac{1}{1000000},$$

$$MTTF = \frac{1}{FailureRate} = \frac{1}{\frac{1}{1000000}} = 1{,}000{,}000(hours)$$

Other Performance Metrics

- MTTF: Mean Time to Failure
  - IF modules have independent, exponentially distributed lifetimes (age of module does not affect probability of failure), the overall failure rate is the sum of failure rates of the modules

Examples:

-- If there are N hard disks, each with MTTF of M hour:

$$FailureRate = N \times \frac{1}{M}, \qquad MTTF = \frac{1}{FailureRate}(hours)$$

-- Further, a system consisting with N_1 hard disks (M_1 hour MTTF per disk), N_2 disk controller (M_2 hour MTTF per controller),, and N_3 power supply (M_3 hour MTTF per supply)

$$FailureRate = N_1 \times \frac{1}{M_1} + N_2 \times \frac{1}{M_2} + N_3 \times \frac{1}{M_3}$$

$$MTTF = \frac{1}{FailureRate} \quad (hours)$$

Other performance Metrics

- MTTF: Mean Time to Failure
  - IF modules have independent, exponentially distributed lifetimes (age of module does not affect probability of failure), the overall failure rate is the sum of failure rates of the modules

Examples:

-- Assuming a system consisting of N disks (M hour MTTF per disk), and the system is considered to fail if X disks fail. Thus:

$$FailureRate = N \times \frac{1}{M} / X \qquad MTTF = \frac{1}{FailureRate}(hours)$$

Basics of Memory Hierarchy

- A key design decision is where blocks (or lines) can be placed in a cache
- **Set Associative**: the set is chosen by the address of the data:

$$(\textit{Block address})\ \mathrm{MOD}\ (\textit{Number of sets in cache})$$

- Caching data that is only read is easy; **Caching writes** is more difficult
- **Cache Miss Rate**: a measure of benefits of different cache organizations
    - Compulsory
    - Capacity
    - Conflict
- **Average memory access time:** better measure of cache organization

$$\textit{Average memory access time} = \textit{Hit time} + \textit{Miss rate} \times \textit{Miss penalty}$$

Miss Rate and AMAT

- As block size increases, so does the miss penalty
- Thus, miss rate does not tell the whole story!
- Average Memory Access Time (AMAT) incorporates miss rate, miss penalty, and hit time

$$\text{AMAT} = \text{Hit time} + \text{Miss rate} * \text{Miss penalty}$$

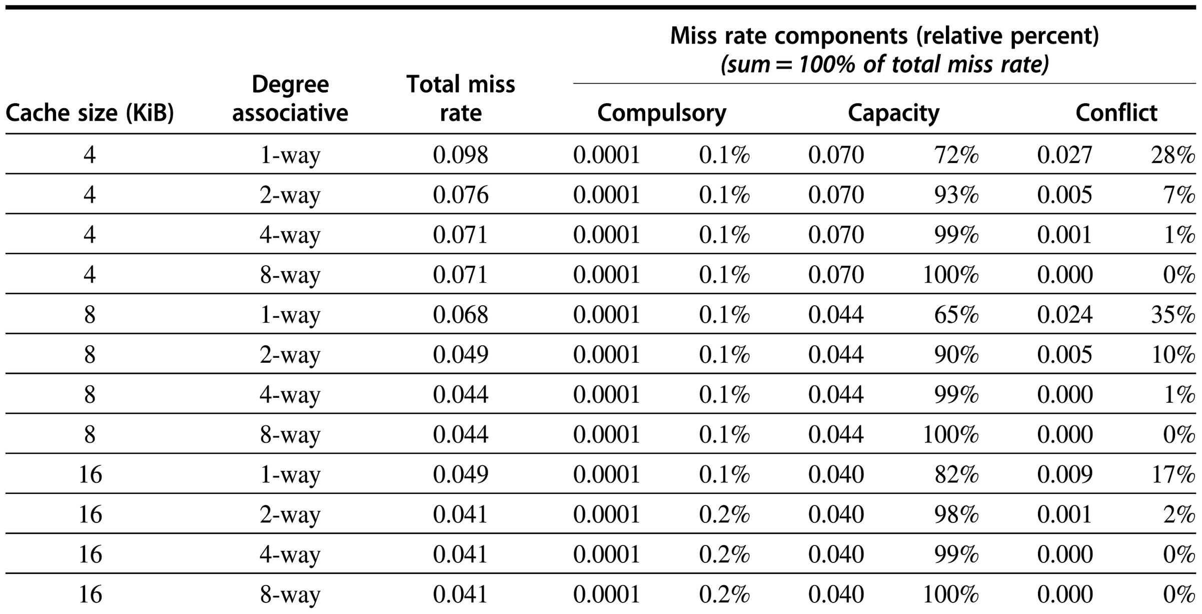
		Cache size			
Block size	Miss penalty	4K	16K	64K	256K
16	82	8.027	4.231	2.673	1.894
32	84	**7.082**	3.411	2.134	1.588
64	88	7.160	**3.323**	**1.933**	**1.449**
128	96	8.469	3.659	1.979	1.470
256	112	11.651	4.685	2.288	1.549

- Average Memory Access Time in clock cycles
- Shows 64 byte as the clear winner for 16kB and greater

The Types of Cache Miss

- The three C's
  - Compulsory – Initial misses due to a cold cache
  - Conflict – Misses due to lack of associativity, i.e. too rigid of a placement strategy
  - Capacity – Misses due to the small cache size

- Mitigation of misses
  - Compulsory misses can be decreased with larger block sizes
  - Conflict misses can be decreased due to associativity
  - Capacity misses can be decreased with larger caches



			Miss rate components (relative percent) *(sum = 100% of total miss rate)*					
Cache size (KiB)	Degree associative	Total miss rate	Compulsory		Capacity		Conflict	
4	1-way	0.098	0.0001	0.1%	0.070	72%	0.027	28%
4	2-way	0.076	0.0001	0.1%	0.070	93%	0.005	7%
4	4-way	0.071	0.0001	0.1%	0.070	99%	0.001	1%
4	8-way	0.071	0.0001	0.1%	0.070	100%	0.000	0%
8	1-way	0.068	0.0001	0.1%	0.044	65%	0.024	35%
8	2-way	0.049	0.0001	0.1%	0.044	90%	0.005	10%
8	4-way	0.044	0.0001	0.1%	0.044	99%	0.000	1%
8	8-way	0.044	0.0001	0.1%	0.044	100%	0.000	0%
16	1-way	0.049	0.0001	0.1%	0.040	82%	0.009	17%
16	2-way	0.041	0.0001	0.2%	0.040	98%	0.001	2%
16	4-way	0.041	0.0001	0.2%	0.040	99%	0.000	0%
16	8-way	0.041	0.0001	0.2%	0.040	100%	0.000	0%

Reducing Miss Penalty

- Reducing Miss Penalty
  - Miss rate is only part of the AMAT equation
  - As the performance gap increased, miss penalties between the L1 cache and memory became prohibitive
  - Multi-level caches solved that problem

- Multi-level Caches
  - Sandwich a larger L2 cache between the L1 and memory
  - Allows L1 to be small to optimize hit time
  - L2 is large enough to have a relatively low miss rate but not too large to keep access time down
  - L2 access time is approximately L1 miss penalty

Miss Rate for L2

- Local miss rate
  - The number of misses divided by the number of access to that particular cache
  - Is often quite low for L2 cache because L1 handles most of the accesses and L2 is only accessed when L1 misses

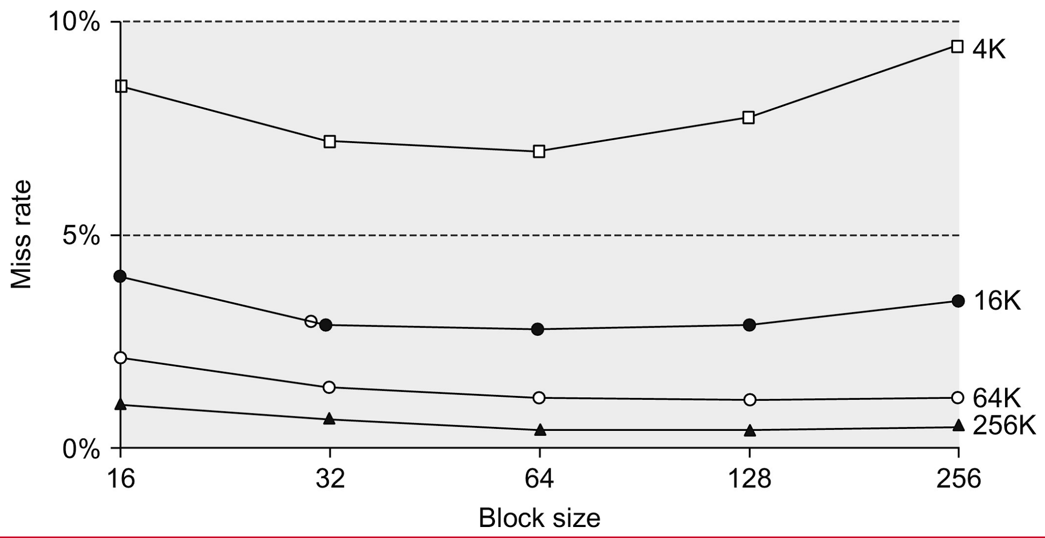
- Global miss rate
  - The number of misses divided by the total number of memory accesses generated by the processor
  - For L1, global miss rate equals the local miss rate
  - For L2, global miss rate equals the local miss rate of L1 times the local miss rate of L2

TECHNIQUES THAT REDUCE MISS RATES

1. Larger Block Size to Reduce Miss Rate

- Larger blocks take advantages of
  - The bandwidth provided by the next level in the hierarchy
  - Spatial locality

- Larger blocks reduce compulsory misses
- Block sized that are too large will start to suffer from conflict misses. *Why*?

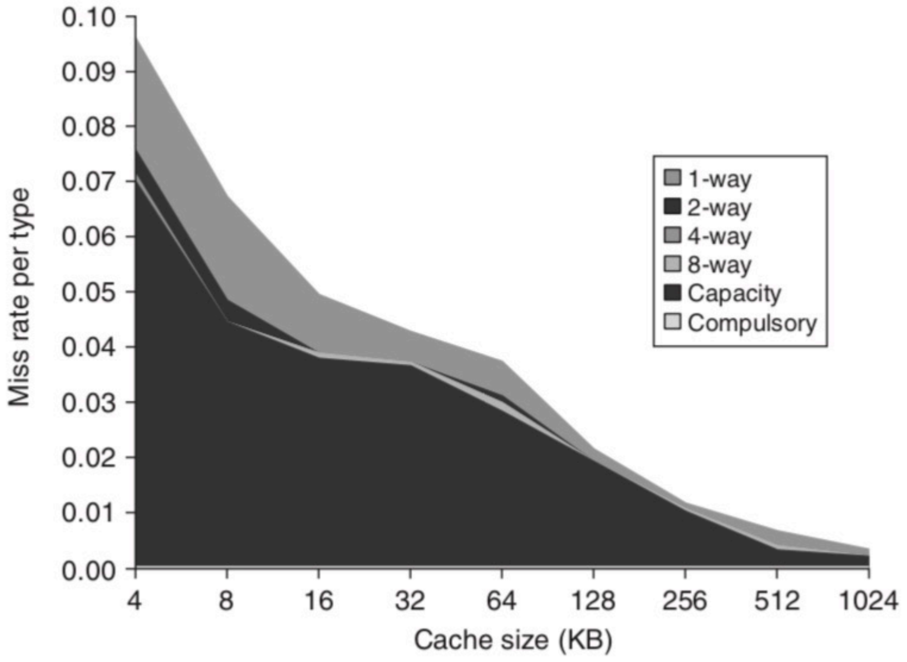


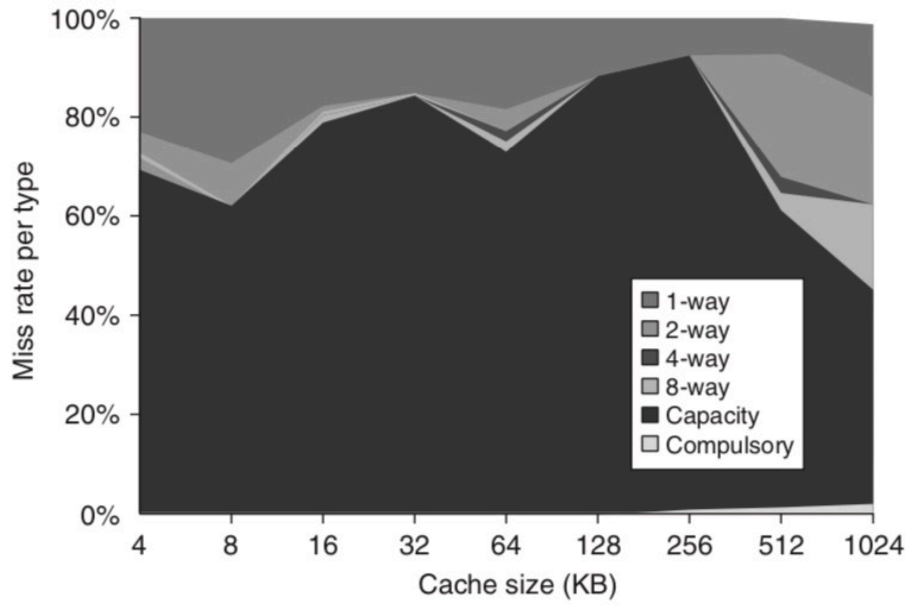
- Block size matters less for larger caches

- 64 bytes seems to be the happy place

TECHNIQUES THAT REDUCE MISS RATES

2. Larger Caches to Reduce Miss Rate



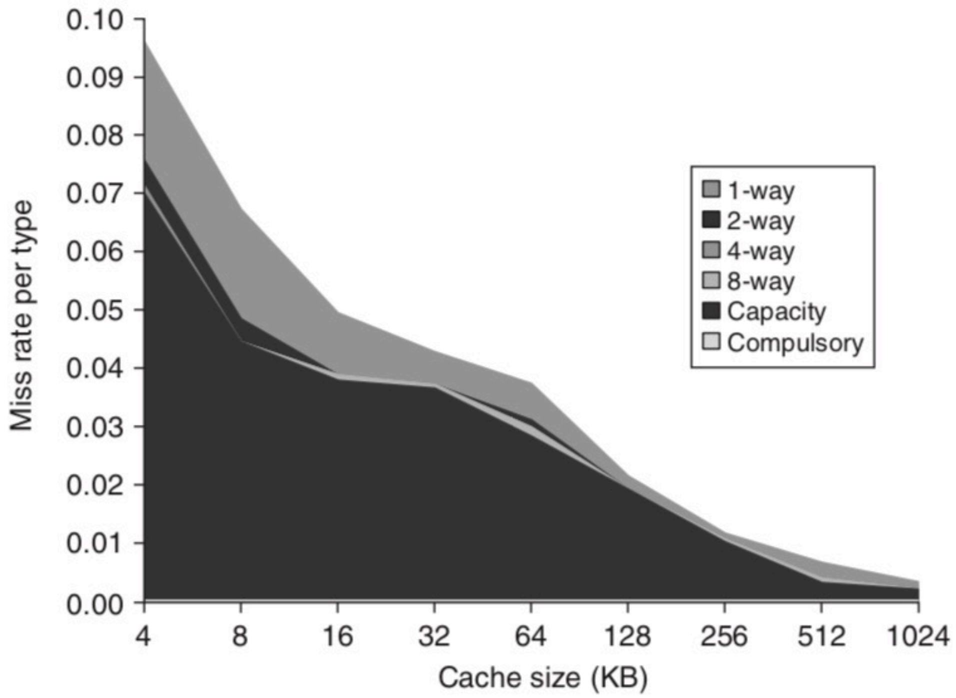


The obvious drawback is potentially longer hit time and higher cost and power. This technique has been especially popular in off-chip caches.

TECHNIQUES THAT REDUCE MISS RATES

3. Higher Associativity to Reduce Miss Rate

- **Observation**: difference by comparing eight-way entries to capacity miss column in this Figure, since capacity misses are calculated using fully associative caches.
- **Observation**: called 2:1 cache rule of thumb, is that a direct-mapped cache of size N has about the same miss rate as a two-way set associative cache of size N/2. This held in three C's figures for cache sizes less than 128 KB



TECHNIQUES THAT REDUCE MISS RATES

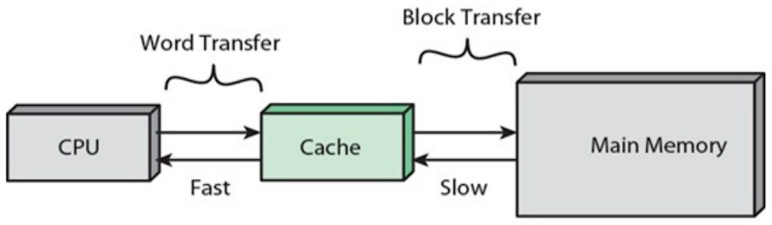
4. Multilevel Caches to Reduce Miss Rate

- Should I make cache faster to keep pace with speed of processors?
- Or make the cache larger to overcome the widening gap between the processor and main memory?

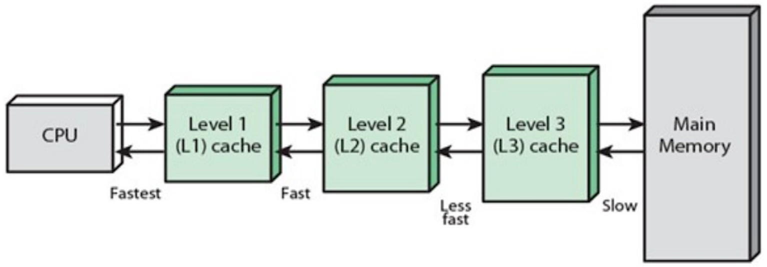
❑ To avoid the ambiguity, these terms are adopted here for a two-level

❖ Local miss rate—This rate is simply the number of misses in a cache divided by the total number of memory accesses to this cache.

❖ Global miss rate—The number of misses in a cache divided by the total number of memory accesses generated by the processor.



Single Cache



Three-level Cache Organization

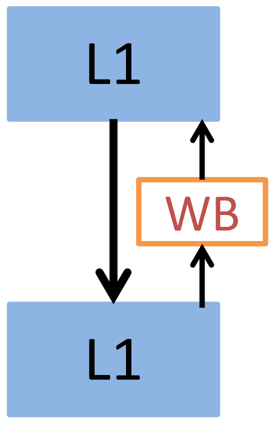
TECHNIQUES THAT REDUCE MISS RATES

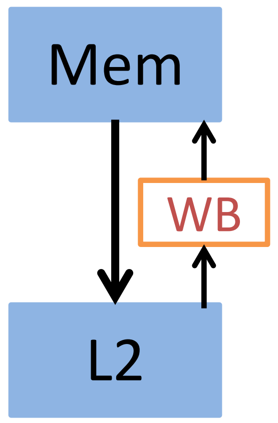
5. Giving priority to read misses over writes to reduce Miss Penalty
6. Avoiding address translation during indexing of cache to reduce Hit Time

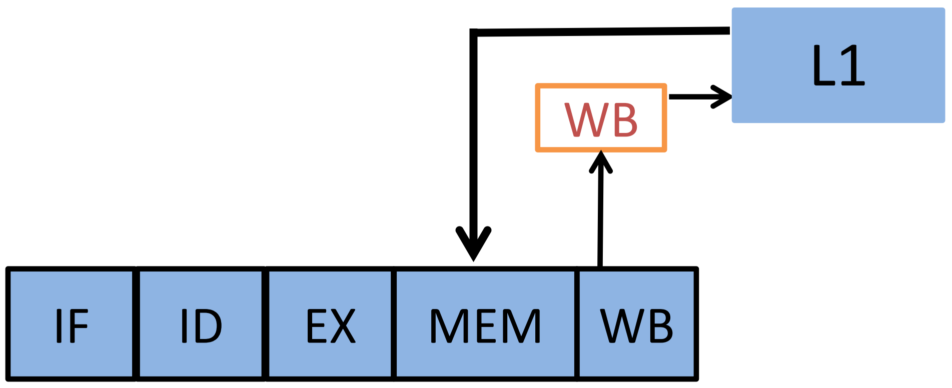
- Virtual Address is not built for all. *WHY?*
  - Protection: Page-level protection
  - Cache is to be flushed
  - Duplicate addresses, called synonyms or aliases
  - I/O typically uses physical addresses

Write buffers to improve Memory performance

- Write Buffers to improve performance of memory hierarchy







MID-TERM EXAM MOCK

1. A Designed 5-stage, pipelined processor and synthesized it for a 45nm process technology node with a target clock rate of 1GHz. During power analysis, the processor is at the target clock rate with a supply voltage of 1.0V. This processor draws 70mW of dynamic power and 10mW of static power. Answer following questions by considering power and energy trade-offs:
   - The energy to complete the operations
   - Scale down the clock and calculate the energy for the operations

(Refer to Lecture 1)

MID-TERM EXAM MOCK

2. MTTF -- Assume a cluster has 500 computers, each of them with a MTTF of 25 days, and the failures follow an exponential distribution and are independent.

- Calculate the MTTF in different situations

(Refer to HW1)

3. Cache misses

- Define and describe the types of cache misses we have discussed in class (Three C's). List other type(s) of cache misses as you know
- Global and local cache miss calculation
- The techniques to improve cache performance

(Refer to Lecture 2 and HW2)

4. Write buffers used in the memory hierarchy

❑ Mid-term Exam

- Date: Oct. 27, 2021 (Wednesday)
- Time: 2:00PM-3:30PM
- In-class
- TA from 2:15pm to about 3:30 or 4pm, in our Lab in the base of ECE building room number L245

Good Luck~